PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 :		(11) International Publication Number:	WO 94/10638
G06F 15/347	A1	(43) International Publication Date:	11 May 1994 (11.05.94)

(21) International Application Number: PCT/AU93/00573

(22) International Filing Date: 5 November 1993 (05.11.93)

(30) Priority data:
PL 5697 5 November 1992 (05.11.92) AU

(71) Applicant *(for all designated States except US)*: THE COMMONWEALTH OF AUSTRALIA [AU/AU]; C/-The Secretary, Department of Defence, Anzac Park West Building, Constitution Avenue, Canberra, ACT 2601 (AU).

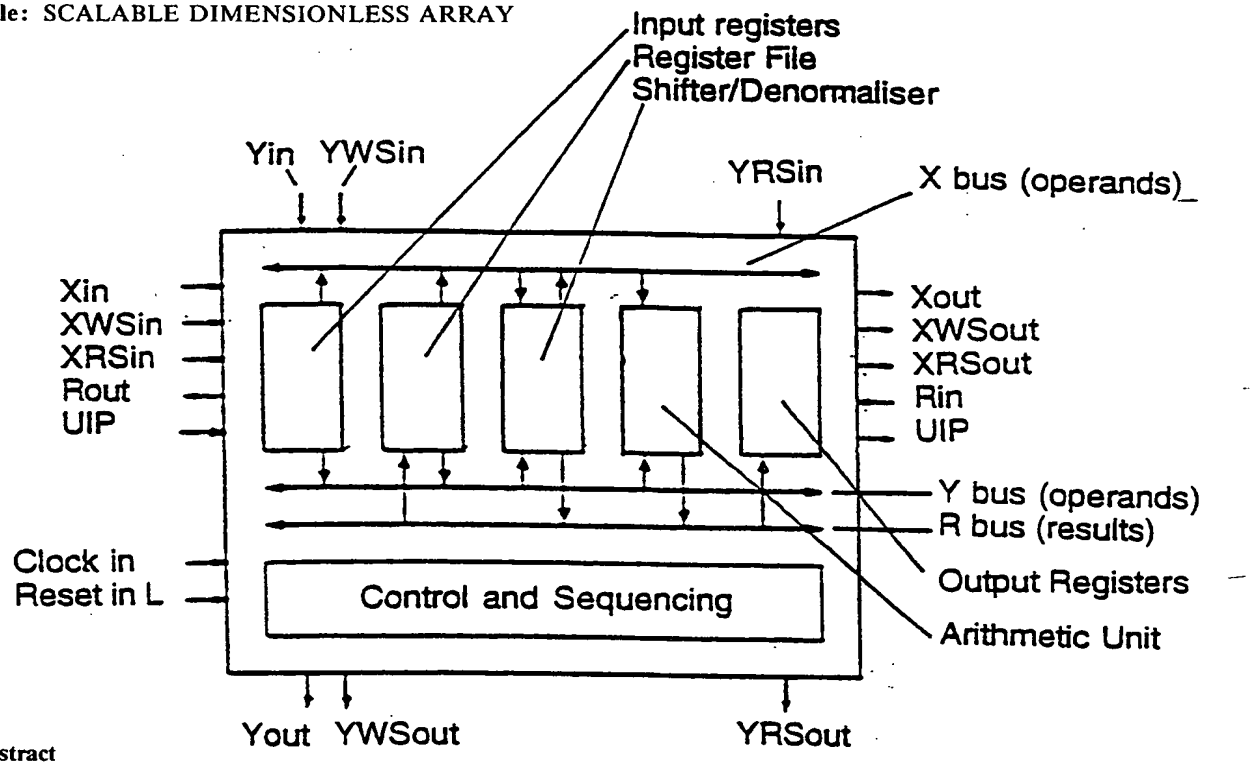
(72) Inventors; and
(75) Inventors/Applicants *(for US only)* : MARWOOD, Warren [AU/AU]; 6 Selangor Avenue, Fairview Park, S.A. 5126 (AU). CLARKE, Allen, Patrick [AU/AU]; 22 Sherbourne Road, Medindie Gardens, S.A. 5081 (AU). CLARKE, Robert, John [AU/AU]; 20 Albert Street, Dulwich, S.A. 5065 (AU).

(74) Agent: COLLISON & CO.; 117 King William Street, Adelaide, S.A. 5000 (AU).

(81) Designated States: AT, AU, BB, BG, BR, BY, CA, CH, CZ, DE, DK, ES, FI, GB, HU, JP, KP, KR, KZ, LK, LU, LV, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SK, UA, US, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: SCALABLE DIMENSIONLESS ARRAY



(57) Abstract

A processing element for use in a scalable array processor chip which can perform a number of point matrix operations for conformable matrices of arbitrary order on an array of fixed size. The processing element includes a number of input and output registers, storage registers, a shifter/normaliser, and arithmetic unit (datapath elements) and a control sequencing unit. The datapath elements are connected by a number of parallel data buses, with the input and output registers connected by serial interfaces.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgystan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo			SE	Sweden
CH	Switzerland	KR	Republic of Korea	SI	Slovenia
CI	Côte d'Ivoire	KZ	Kazakhstan	SK	Slovakia
CM	Cameroon	LI	Liechtenstein	SN	Senegal
CN	China	LK	Sri Lanka	TD	Chad
CS	Czechoslovakia	LU	Luxembourg	TG	Togo
CZ	Czech Republic	LV	Latvia	TJ	Tajikistan
DE	Germany	MC	Monaco	TT	Trinidad and Tobago
DK	Denmark	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	US	United States of America
FI	Finland	ML	Mali	UZ	Uzbekistan
FR	France	MN	Mongolia	VN	Viet Nam
GA	Gabon				

SCALABLE DIMENSIONLESS ARRAY

TECHNICAL FIELD

This invention relates to the general field of digital computing and in particular to a scalable array of globally clocked multiply/accumulate floating point processing elements.

BACKGROUND ART

Kung and Leiserson ['Systolic Arrays (for VLSI)' in Sparse Matrix Proceeding 1978, Soc. for Industrial and Applied Mathematics, 1979] presented the concept of performing matrix operations using arrays of simple processing elements. Each processing element implements a simple primitive operation. As an example, at a given time, a processor may:

read the input data vector {a(in),b(in),c(in)},
perform an arithmetic operation such as c(out) = a(in)b(in) + c(in),
write the output data vector {a(out),b(out),c(out)}.

The processing elements are connected only to their nearest neighbours, and so the problems of routing, fan-out and clock skew are minimised. Data and results move synchronously through the array of elements. The name applied to this approach to computation with arrays of identical processing elements is systolic.

An example algorithm quoted by Kung and Leiserson was the matrix product. Using a systolic array in which the processing elements executed the local algorithm presented above, known as an inner-product-step algorithm, they showed that the system level algorithm which this implemented was a matrix product of computational order O(N), rather than the computational order O(N3) for the matrix product implemented on a conventional scalar architecture. The matrix product is represented simply as

$$C = AB$$

where A, B and C are matrices of a size equal to the order of the array.

Conformal matrices can be multiplied with this array under certain restrictions if the results are re-circulated through the array, and larger order matrix products can be computed if the task is partitioned. Speiser, Whitehouse and Bromley ['Signal Processing Applications for Systolic Arrays', Record of the 14th Asilomar Conference on Circuits, Systems and Computers, IEEE No. 80CH1625-3, 1980] subsequently demonstrated the use of inner-product-accumulate processing elements for the same matrix product algorithm. In this case, the results are formed in-place, and do not move between processing elements. The only difference between the description of this algorithm and the algorithm described above is that the input and output phases of the algorithm do not include the reading and writing of c(in) and c(out) respectively, and that an explicit unload phase must be added at the end of the algorithm to return the results.

The primary advantage of systolic processing over conventional linear processing is speed. The systolic architecture uses the fact that for matrix multiplication, the same operand data may be reused many times in the computation of cross-product terms, thereby making better use of the available data bandwith. The improved performance, however, comes at the cost of flexibility. Prior art devices have been designed for very specific applications such as Fast Fourier Transform computations or video signal processing. An advantage of the present invention is the ability of the same device to be useful for a wide variety of matrix computations without the need for hardware reconfiguration. The device is particularly useful when implemented as an architectural enhancement to a computer in which case the processing power of the computer is considerably enhanced.

DISCLOSURE OF THE INVENTION

It is an object of this invention to provide a processing element for use in a scalable array processor which is able to implement a set of primitive floating point matrix operations for conformable matrices of arbitrary order on an array of fixed size.

It is a further object to provide a scalable array processor chip which is able to perform one or more of the following functions :

compute the product of two matrices

compute the element-wise (Hadamard) product of two matrices

compute the sum of two matrices
permute the rows and columns of a matrix
transpose a matrix.

It is a still further object of this invention to at least provide the public with a useful alternative to existing systolic devices.

Therefore, according to perhaps one form of this invention, although this need not be the only or indeed the broadest form, there is proposed a processing element suitable for use in a scalable array processor comprising :
at least one input register means adapted to receive and process serial operands in the form of {instruction, data} 2-tuples;
a memory means adapted to store temporary results and constants;
a computing means adapted to perform logical operations;
an output register means adapted to output results from the processing element;
a control and sequencing means adapted to control the operation of the processing element;
a plurality of data buses adapted to provide communication between the plurality of means.

In preference the computing logical means consists of a shifter/normaliser means adapted to shift/normalise data and an arithmetic means adapted to perform logical operations such as but not limited to addition, subtraction and partial multiplication operations.

In preference the processing element is adapted to perform floating point multiply, floating point add and floating point multiply-accumulate which is used for inner product accumulate operations.

In preference the input register is adapted to output a copy of the input operand bit with a one clock period delay.

In preference there are N input registers and the processing element is suitable for use in a N-dimensional scalable array processor.

In preference N can be any positive integer.

In preference the input registers convert the input serial data to an internal representation comprising separate sign, fraction and exponent.

In preference the memory means consists of read only memory for storage of constants and a read/write memory for storage of temporary results.

In preference the shifter/normalizer means is adapted to perform binary weighted barrel shifting wherein the shifter function is determined by a control input to the shifter/normalizer and the normalizer function effects a data dependent shift of up to 15 bits within a single clock cycle.

In preference the arithmetic means implements logical operations such as but not limited to floating point addition, multiplication and multiply-accumulate algorithms using a parallel microcoded data path.

In preference the arithmetic means comprises a logical unit such as but not limited to an input-multiplexer, an adder, an output shifter, flags unit and a control unit.

In preference the output register can be loaded in parts to enable the conversion from the internal representation to IEEE 754 floating point format. The output register can be parallel loaded from the arithmetic means or can be serially loaded from a serial source. The register is unloaded serially.

In preference the control and sequencing means includes timing and control logic, a microcode ROM, address decoders, branch control logic, flags logic, instruction register, instruction decoder and a program counter.

In preference there are three data buses, an X bus, a Y bus and a R bus. The X and Y buses are called operand buses and the R bus is called the result bus.

In preference the processing element has an accumulator comparison means.

In another form the invention consists of a scalable array processor chip comprising an array of processing elements each said element including :
at least one input register means adapted to receive and process serial operands in the form of {instruction, data} 2-tuples;
a memory means adapted to store temporary results and constants;

a shifter/normalizer means adapted to shift or normalize data;
an arithmetic means adapted to perform logical operations such as but not limited to addition, subtraction and partial multiplication operations;
an output register means adapted to output results from the processing element;
a control and sequencing means adapted to control the operation of the processing element;
a plurality of data buses adapted to provide communication between the plurality of means; and
wherein each processing element has means for communication only with adjacent elements.

In preference the array of elements comprise an interconnected lattice of at least one dimension.

In preference the array of elements comprise an interconnected lattice of at least two dimensions.

In preference the scalable array processor chip is adapted to perform at least the functions of computing the product of two or more matrices, computing the element-wise product of two or more matrices, computing the sum of two or more matrices, permuting the rows and columns of a matrix and transposing a matrix.

In a yet further form of the invention there is proposed a computing apparatus comprising a host processor, at least one scalable array processor chip and a plurality of data formatters wherein the scalable array processor chip(s) and plurality of data formatters are adapted to perform matrix operations otherwise performed by the host processor.

In preference the apparatus includes a memory cache adapted to store operand data and temporary or intermediate results.

In a still further form of the invention there is proposed a method of performing matrix operations comprising the steps of :

(a) providing a plurality of processing elements in the form of an array adapted to perform systolic processing operations;

(b) receiving operand matrix data for processing from a host or data source;

(c) formatting the operand matrix data in a data formatter by adding an instruction to form an {instruction, data} 2-tuple;

(d) transferring sets of 2-tuples to the processing element array to cause the processing elements to process the data in accordance with the instruction;

(e) repeating the steps (b) to (d) a number of times said number of times being dictated by the matrix operation being performed;

(f) unloading the results of the matrix operations into an output result register of the processing elements (under control of an instruction specified by the operand 2-tuple);

(g) transferring the contents of the output result registers held within the plurality of processing elements back to data formatters as result wavefronts;

(h) storing the result wavefront data back to a host or data sink; and

(i) repeating the steps (f) to (h) a number of times said number of times being dictated by the matrix operation being performed.

In preference, the data formatter is of the type the subject of co-pending patent application number PL5696 entitled "DATA FORMATTER".

The sets of 2-tuples are known as operand wavefronts. During the unload step, an indication is provided to the data formatter that the unload operation is occurring, allowing synchronization of data transfers to and from the processing array.

During step (g), the results are transmitted in sets containing one result from each of the processing elements at the left edge of the array. Such a set is known as a result wavefront.

There may be as many result wavefronts held within the array as there are columns of processing elements.

BRIEF DESCRIPTION OF THE DRAWINGS

For a better understanding of this invention a preferred embodiment will now be described with reference to the attached drawings in which :

FIG. 1 is a schematic diagram of one processing element;

FIG. 2 is a schematic diagram of one embodiment of a systolic array

processing element chip utilising the elements of FIG. 1;

FIG. 3 is a schematic diagram of a first embodiment of a processing apparatus utilizing the chip of FIG. 2;

FIG. 4 is a schematic diagram of an inner-product-step processor;

FIG. 5 is a schematic diagram of an inner-product-accumulate processor;

FIG. 6 is a schematic example of the entry of operand wavefronts to a processor array;

FIG. 7 is a schematic example of the unloading of result wavefronts from a processor array;

FIG. 8 is a schematic example of the entry of element-wise operand wavefronts to a processor array;

FIG. 9 is a schematic example of the unloading of element-wise result wavefronts from a processor array; and

FIG. 10m is a schematic diagram of a second embodiment of a processing apparatus.

BEST MODE FOR CARRYING OUT THE INVENTION

Referring now to the drawings in detail, each processing element consists of a number of input registers, a memory consisting of a register file and a constant ROM, a shifter/normalizer, an arithmetic unit, output registers and a control and sequencing unit.

The datapath elements (input registers, memory, shifter/normalizer, arithmetic unit and output registers) are interconnected by three parallel data buses. In addition serial interfaces are provided to and from each of the input registers and the output register to allow communication between processing elements and to facilitate construction of arbitrarily large arrays of processing elements.

An array computes 2N2 floating point operations (1 multiply and 1 accumulate

for each processing element in the array) in the time taken to fetch 2N operands. To ensure that the computation is bandwidth limited, each processing element needs only compute at a rate of one floating point operation every N data fetches. This fact leads to the conclusion that very cheap processing elements can be used in the array.

A schematic of the processing element is shown in FIG. 1. The choice of a simple microcoded datapath and sequential algorithms to perform the floating point operations means that the size of the processing element can be kept small. Many such processing elements can therefore be placed on a single chip. The fact that processing elements implemented in this manner are slower than those built using fully parallel algorithms and architectures becomes insignificant as the size of the array is increased. This is because the processing performance achieved is limited by the data bandwidth (and array size), not by the computation rate for a single processing element.

The functions performed by each module in the processing element are described below.

Input Registers: The input registers receive serial operands in the form of {instruction, data} 2-tuples from adjacent processing elements to the left or top, or in the case of processing elements at the top or left boundary of the array, from operand data formatters. They then separate the instruction and reformat the data to an internal representation consisting of separate sign, exponent and fraction words. This data is available to the processing element via the X and Y internal data buses. The input registers also compute the sign of the product of the two inputs, check for zero operand data and implement the Booth encoder used during multiplication operations.

Memory: The memory consists of a Register File and a Constant ROM: The register file is a 5 word memory used to hold the product (both fraction and exponent), accumulator (both fraction and exponent) and temporary results. The product and accumulator registers can be swapped under the control of microcode to facilitate efficient implementation of the pre-alignment operation in the floating point addition and accumulation algorithms. The registers can be loaded from the R bus, and their contents can be read from either the X or Y buses. The Constant ROM stores a number of constants that are used during the implementation of the floating point algorithms. These can be read via the X

and Y operand data buses.

Shifter/Normalizer: Under microcode control, the shifter can either operate as a shifter (for pre-alignment of fractions before addition) or a normalizer. When acting as a shifter, it performs right-shift operations on one operand datum (The X operand). The amount by which the datum is shifted is determined by a previously computed shift that is applied to the second input to the shifter (the Y operand). The shifter can shift 0 to 15 bits right within one clock cycle. When acting as a normalizer, the Shifter/Normalizer performs either a right shift by one bit, or a left shift by 0 to 15 bits within one cycle. In this case the shift is applied to the X operand input to the shifter and is independent of the Y operand input. The value of the shift is data dependent. A right-shift is performed if the value on the X input is the result of a computation which had overflowed (such as in the case of addition of two normalized numbers having the same exponent). Otherwise, a left-shift is performed. When acting as a normalizer, the Shifter/Normalizer at the same time computes the offset (exponent offset) that must be applied to the exponent of the number being normalized in order to compensate for the shift that is applied. Shifting and normalization operations that require shifts of greater than 51 bits can be implemented by multiple passes through the shifter/normalizer.

Arithmetic Unit: The arithmetic unit consists of an input multiplexer, an adder, a result shifter and a flags unit. There are two parallel data inputs (X and Y) to the arithmetic unit and a single parallel data output (R). The input multiplexer can be used to complement and/or left-shift the X operand under control of the Booth encoding logic contained in the input registers. This feature is used in the implementation of multiplication using a modified Booth algorithm. The multiplexer can also be controlled directly by the processing element's microcode to facilitate the implementation of addition, subtraction and data-move operations.

The adder performs conventional two's complement addition. The carry input to the adder can be controlled by either the booth encoder logic or the processing element's microcode. Both addition and subtraction can be performed by the combination of input-multiplexer and adder. Under control of the processing element's microcode, the output shifter latches either the result of the computation or the result divided by 4. This feature is used during partial multiplication operations. The latched result remains valid until the next time the

arithmetic unit is used. The latched result can be written onto the result bus R. A number of flags are set or cleared depending upon the result latched by the arithmetic unit's output shifter. These flags include the sign of the result, whether or not the result is zero, and whether or not the result is less than or equal to 15 (used to support multi-pass shifting during addition pre-alignment).

Output Registers: The output register module is used to communicate the results of computations back from the processing element toward the left boundary of an array of processing elements. The output register can be parallel loaded from the arithmetic unit or can be serially loaded from a serial source (often the serial source is from another processing element's output register). The output register is unloaded serially. The output register is parallel loadable by the arithmetic unit in three parts: sign, exponent and fraction. This facilitates conversion from the internal data representation to IEEE 754 floating point format.

During the time when the arithmetic unit is converting the accumulator contents into IEEE floating point format, a flag is set to indicate that a register unload is in progress (UIP).

Control and Sequencing: This module includes a microcode ROM, a program counter, branch control logic, flags logic, an instruction register, an instruction decoder, address decoders and timing and control logic. This circuitry is used to sequence the processing element through its operations. Each clock cycle, the microcode ROM issues a microinstruction to the processing element's datapath units, and thereby controls the function and timing of the data operations being performed. Data and control flags fed to the branch control logic enable the processing element to perform data dependent operations required for implementation of the floating point algorithms. Fields of the instruction transmitted serially to the processing element as part of the {instruction, data} 2-tuple are also fed to the branch control logic and flags logic of the Control and Sequencing Unit. These also determine the sequence of microinstructions executed by the processing element. The instructions specified in the {instruction, data} 2-tuple are distinct from the set of microinstructions implemented by the processing element The instructions specified in the {instruction, data} 2-tuple control the flow of execution of the processing element's microcode.

The internal data representation used by the processing element uses two 32-bit data words to represent each IEEE single precision number. One of the two words represents the mantissa in 2's complement form, normalized to bit 29. The second word represents the exponent using an exponent bias of 229. This format provides better resolution in the mantissa than IEEE single precision format, and the use of a large exponent field virtually guarantees that exponent overflow cannot occur.

Within each processing element, multiplication is facilitated by the inclusion of a modified Booth encoder and multiplexer. The denormalisation and normalisation operations required by the floating point accumulation or addition algorithms are facilitated by the repeated application of the shifter circuit which can shift up to 15 bits in a single cycle.

FIG. 2 shows a scalable array processing chip composed of a 5 x 4 rectangular array of single precision floating point processing elements which accept serial dataflow operands, and which perform a set of operations on those operands. Each operand consists of a 5-bit instruction followed by an IEEE standard single precision number. Each processing element is a microcoded ALU with a 32-bit parallel datapath that includes dedicated hardware support for floating point multiplication and addition algorithms.

The array of processing elements is clocked synchronously. The three bit-serial links provide communication between processing elements. One link is provided for each of the two input X and Y operands and one for the output, or result operand R. As shown in FIG. 2, input data is transferred from left to right across the array, and output results are transmitted from right to left. Chips can be cascaded arbitrarily in both X and Y directions.

The operation of the scalable array processing chip is described with reference to the system block diagram shown in FIG. 3. The data interface provides communication between the scalable array chip and the host system. The data formatter elements are described separately in a co-pending application number PL5696 entitled DATA FORMATTER.

The I/O architecture of each processing element consists of two orthoganal data transmission paths for X and Y operands, each consisting of a single one-bit delay cell and a 32-bit data storage register. The X operand path also

includes a 5 bit instruction register. Data is input to the array as a sequence of {instruction,data} 2-tuples. These are split into separate instruction and data words on receipt by the input registers.

Each X data operand consists of a 5-bit instruction followed by a single 32-bit IEEE 754 standard floating point number. A variable length gap of several clock periods may be present between operands for I/O synchronisation. The operand is transmitted in bit serial form into the processing element. When the entire {instruction, data} 2-tuple is held within the processing element, it is cross-loaded into parallel holding registers. The instruction is decoded and used to control the execution of the floating point algorithms. The data is converted by hardware into the internal extended format. The internal format has both extended precision and extended dynamic range when compared with the IEEE standard.

The bit-serial data is bit-skewed on entry to adjacent processing elements on the array boundary. This skew is preserved between adjacent elements within the array by passing the data through the single-bit delay stage in each processing element before re-transmitting it to the next processing element. The use of serial data both minimises the I/O pin count at the array boundary and allows adjacent processing elements to both commence and conclude their computations with a time differential of only one bit period. The advantage of the bit-skewing approach over a broadcast architecture is that there is no need to drive long buses with large buffers and thereby provides the capability for arbitrary expansion of the array.

Bit skewing has the advantage over word-skewing in that less wavefronts are required to complete a processing task. The bit-skewed approach therefore results in the minimisation of job time. The computation time is minimised for both a single job and a job stream.

At the completion of a set of computations, an operand wavefront is issued to the array which causes the unloading of the results into the output registers of the processing elements.

Clocking of the scalable array processing chips is performed by a single phase 50% duty cycle clock from which all internal timing signals are generated. The clock is buffered on entry to the chip and is distributed to each processing

element. It is re-buffered within the processing element where it is used as a locally synchronous clock. In addition, each processing element generates a second, synchronous clock of the same frequency but with a duty cycle determined by a self-timed circuit. The secondary clock is used to provide timing information for bus precharging, data transfers and evaluation of execution units.

FIG. 4 shows schematically the inner-product-step process described by Kung and Leiserson. Data is clocked into each processing cell from the left and top edges while the results are clocked out from right to left. For a matrix product algorithm, an inner product accumulate algorithm is used in preference to the inner product step process common in much of the prior art. The inner-product-accumulate process is depicted schematically in FIG. 5. Data is again clocked into the element from the left and top but in this case the result is formed in place. An explicit unload phase is implemented to obtain the result after the computation is complete. An advantage of the inner product accumulate algorithm over the inner product step approach is illustrated when matrix products are computed for matrix operands which are rectangular. The inner product step process requires the recirculation of the result partial product matrix. In contrast, the inner product accumulate algorithm computes the result in-place, and incurs no hardware penalties, irrespective of the length of the inner products.

The sequence of operations performed by the processing elements is determined by the 5 bit instruction transmitted as part of the X operand. The five instruction fields and their function are listed in the table below.

Instruction	Bit No.	Function
ADD	4	Floating point add
LDR	3	Convert result to IEE format and load O/P register
HAD	2	Enable result unloading only if active flag set
SDE	1	Set active flag if accumulator contents are non-zero
CLR	0	Clear accumulator prior to computation

TABLE 1

The default operation performed by the PE (i.e., when none of the fields of the instruction are asserted) is an inner-product operation implemented as a floating point multiply-accumulate, the input X and Y operands being multiplied and accumulated with the contents of the accumulator.

If the CLR field is asserted, the accumulator contents is cleared before the computation commences. This generally occurs for the first wavefront of a matrix multiplication, and also when executing element-wise operations. The accumulator is cleared before the computation is commenced but after the ACTIVE flag is set if (the SDE field) is set, and after the accumulator has been unloaded into the result register (if the LDR field is set).

If the SDE field is set AND the value held in the accumulator (from the previous operation) is non-zero, an internal flag, ACTIVE (one per processing element) is set to indicate that this processing element is an active element. Only active elements are permitted to unload results during element-wise operations.

If the HAD field is asserted, the operation being performed is deemed to be an element-wise (hadamard) operation. If this field is set, only those processing elements flagged as active elements (as determined by their ACTIVE flags) can unload their accumulator contents into their output register R.

If the LDR field is asserted, the accumulator contents from the previous computation are converted back to IEEE format and are unloaded into the processing element's output register.

During the unloading process, the processing element issues a flag (UIP) to indicate that the unload is in progress.

If the ADD field is asserted, the X and Y operands are added rather than multiplied prior to the result being stored in the accumulator. The HAD and CLR fields must also be asserted for matrix addition instructions.

The element-wise operations of addition and multiplication defined by

$C = A + B$ where $c_{ij} = a_{ij} + b_{ij}$

$C = A \bullet B$ where $c_{ij} = a_{ij}b_{ij}$

are performed by first setting the active-element (ACTIVE) flag in a desired processing element. This procedure is typically done once during system initialization. It is achieved by issuing an instruction with the SDE (set active element) field asserted. When this occurs, processing elements that contain non-zero results in their accumulators set the value of their ACTIVE flag to TRUE.

The processing elements accept operand data and return results in IEEE standard format. Internally, an extended precision format is used for both the mantissa and exponent of the partial results.

The internal formats used for the representation of mantissa and exponent are as follows:

2's complement mantissa	sgf.fffffffffffffffffffffffffffff
Exponent	0eeeeeeeeeeeeeeeeeeeeeeeeeeeeee
Exponent Bias	00100000000000000000000000000000

where:

s is the sign bit of the mantissa. +/ is represented as 0/1 respectively.

g is a guard bit used to avoid mantissa overflow during accumulation.

f is a fraction (mantissa) bit. The mantissa is normalized : the most significant fraction bit is 1 (explicit).

. is the position of the binary point (showing that the mantissa is normalized).

e is a bit of the exponent, which is held in biased form. The exponent bias is 229.

If the flags in the anti-active processing elements have been set by a prior SDE instruction, and an element-wise multiplication of a matrix A with the unit matrix is executed, the result of the operation is the transpose of the matrix A. If an arbitrary orthogonal set of elements have their flags set, a permutation of the input matrix will be performed by this element-wise product.

When an unload (LDR) instruction is received, the accumulator contents are converted from the internal format to an IEEE standard form. Numbers outside

the range that can be represented by the IEEE single precision format are truncated to zero (in the case of results with large negative exponents, including IEEE denormalized numbers) or limited to infinity (in the case of numbers with large positive exponents). In both cases, the sign of the zeros or infinities are retained (unless the result is a true zero, in which case positive zero is always returned).

The IEEE representation of the result is loaded into a separate output register which is concatenated with other output registers in adjacent processing elements to form an output register chain. The result is output in a serial form through this register chain.

Matrix algorithms which are elements of the set of primitive operators {multiplication, addition, element-wise (or Hadamard) multiplication, permutation} are performed directly by the processing array. Implementation of these operations for operands whose dimension exceeds the size of the array is possible by mathematically partitioning the operations to a set of operations which can be computed separately using the available array size.

For the particular case where the problem size does not exceed the size of the array, recursive algorithms can be implemented which recirculate the output of the array back to its input. This can be a useful method to minimise memory bandwidth requirements in particular applications.

If a matrix multiplication is commenced with an instruction which does not clear the accumulator, the result of the multiplication will be summed with the prior result. This gives a matrix multiplication/accumulation capability which has direct application to the evaluation of complex matrix operations.

FIG. 6 shows the way in which conformal matrix operands are entered into the systolic array. Bit-skewing is indicated by the small offset between adjacent rows of A and columns of B. Each element of the processor array computes an element cij of the result matrix C, by evaluating the inner product $C_{ij} = \sum_{k=0}^{N} a_{ik} b_{kj}$. When the last wavefront has been input to the array, the result matrix may be read from the array. The elements are obtained in the order shown in FIG. 7.

If the processing elements on the main diagonal in the array have their active

element flag set by an arbitrary prior operation as shown in FIG. 8, the processor array can be used for the element-wise operations of addition and Hadamard multiplication. FIG. 8 shows the entry of conformal matrices to a 4 x 4 subarray of the chip for the purposes of element-wise addition or multiplication. Only those elements shown as • are used.

FIG. 9 shows the relationship between rows of data which are output from the array after an element-wise operation. Due to the word-length registers present in the output register chain, the data is skewed by one word-time plus one bit-time. The additional bit-time delay is caused by the bit skewing of the input operands.

In a second embodiment the invention has been implemented in a system hosted by a Sun SPARCstation. The matrix processor is interfaced to the Sun SPARCstation via the SBus. This arrangement is convenient since it allows the SCAP hardware to operate using virtual addressing, with virtual to physical translation being performed by the SBus controller in the SPARCstation. Th host processor and the matrix processor therefore share the same data space, so both can interact with the matrix data directly. This approach does however have its own disadvantages, the most critical being the fact that the data transfer rate across the SBus tends to be quite low due to the overheads of address translation.

To compensate for this low data rate, the matrix processor also includes a cache memory subsystem. The cache supports burst mode data transfers across the SBus on cache misses and can also be used to hold frequently used operand matrices (such as coefficient matrices in transform applications) and to store temporary or intermediate results.

A novel cache partitioning scheme has been implemented. The technique allows the cache to be dynamically divided into a number of regions that are guaranteed not to interact thereby ensuring that fetches for one matrix operand do not interfere with fetches for the other. The data controllers determine how the cache is partitioned on a per-operand/result basis (it is also possible to assign a cache partition to the instruction streams) by issuing an 8-bit space address along with each address generated. Each bit of the space address can be set or cleared, or can take on the value of one of the generated address bits. In our system implementation, three bits of this space address are used to

control non-cached accesses, temporary matrix accesses and temporary matrix initialization. Four bits are used to partition the cache into up-to 16 independent regions. Use of the temporary matrix control bits of the space address allows temporary result matrices to be stored entirely within the cache without being written out to the host. In fact, such matrices are entirely invisible to the host processor. The maximum data throughput obtainable using the cache is 12.5 Mwords/second.

The two custom chips implemented during the development of this system are a processing element array chip and a data controller chip. Both chips were designed using a generic 1.2 micron double layer metal CMOS process rule-set and were retargetted for fabrication using a 1.0 micron process using a gate shrink.

The processing element array chips are full custom integrated circuits each containing an array of 4 rows by 5 columns of floating point processing elements. Because the overall computation rate is limited by the available data bandwidth, the speed of computation of the processing elements if not overly important. Therefore, the architecture has been designed to yield processing elements (PEs) that are physically small rather than being particularly fast. Each complete floating point unit occupies only 2.7sq mm.

The processing element does not include a dedicated hardware multiplier, but is implemented as a simple microprogrammed 32-bit datapath with hardware support to aid the floating point computations, as illustrated in FIG. 5.

The PE hardware incorporates a booth encoder and multiplexer to facilitate multiplication using an iterative modified booth algorithm, and also a shifter/normalizer that can be used for pre-addition alignment as well as post addition normalization. When used as a normalizer, the shifter has the ability to compute the amount by which the exponent must be adjusted during the same time that the normalization occurs. Computation of the floating point arithmetic operations (multiply/accumulate, multiply or add/subtract) are completed within 40 clock cycles.

The processing element array chip accepts IEEE single precision floating point numbers as inputs and feeds results back through the data controllers in the same format. Internally, a proprietary number representation is used, including

a 31 bit exponent that virtually eliminates the possibility of exponent overflow.

The chips operate at 20MHz clock speed, achieving around 20 MFLOPS peak performance per chip. Processing arrays of arbitrary size can be built with no external components simply by stacking the chips to form a two dimensional array. The pin-out of the chip is such that 1-to-1 connection of inputs and outputs of adjacent chips can be made. All communication to and from the array is via the edge elements of the array. Operand data enters the array on left and top edges. This data is known as the X and Y operand data respectively. The result data (R) emerges from the left edge of the array and can be extracted independently from the application of operand wavefronts (that is, the operand and result streams operate in parallel).

The only global signals in the array are clock and reset. Because all communication is local (nearest neighbour only), the system is insensitive to clock skew from one side of the array to the other. The only requirement is that the skew between adjacent PEs is kept under control. This can be readily achieved by orderly layout of clock routing and/or insertion of clock buffering.

The processing elements are low power devices due to their architecture. The entire chip containing 20 processing elements dissipates less than half a watt. This corresponds to less than 5mA per processing element at 20MHz operation, or 5mA per MFLOP.

Number of Transistors	270 000
Die Size (Pad to Pad)	8.56mm x 8.35mm
Transistor Density	3800 T/sq mm
Power Dissipation	0.5 Watts
Package	68 CLCC
Floating-point Performance	20 MFLOPS @ 20 MHz
Design Style	Full Custom

TABLE 2

The performance attained by the apparatus of the second embodiment for a range of applications is shown in Table 3.

Application	Exec. Time	Performance
3450 point 1D Fourier transform using 2D factorization	20 msec	130 MFLOPS
2D Fourier Transform of 380380 point image	1385 msec	66 MFLOPS
4000 tap FIR Filter	35 msec per 1000 data samples	210 MFLOPS
10th order Matrix polynomial evaluation of 60 x 60 complex matrix	136 msec	114 MFLOPS
QR factorization of 59 x 60 Matrix	561 msec	87 MFLOPS

TABLE 3

1. A processing element suitable for use in a scalable array processor comprising of at least one input register means adapted to receive and process serial operands in the form of {instruction, data} 2-tuples, a memory means adapted to store temporary results and constants, a computing means adapted to perform logical operations, an output register means adapted to output results from the processing element, a control and sequencing means adapted to control the operation of the processing element;
a plurality of data buses adapted to provide communication between the plurality of means.

2. An apparatus as in claim 1 wherein the computing logical means consists of a shifter/normaliser means adapted to shift/normalise data and an arithmetic means adapted to perform logical operations such as but not limited to addition, subtraction and partial multiplication operations.

3. An apparatus as in claim 1 wherein the processing element is adapted to perform floating point multiply, floating point add and floating point multiply-accumulate which can be used for inner product accumulate operations.

4. An apparatus as in claim 1 wherein the input register is adapted to output a copy of an input operand bit with a one clock period delay.

5. An apparatus as in claim 1 wherein there are N input registers and the processing element is suitable for use in a N-dimensional scalable array processor.

6. An apparatus as in claim 5 wherein N can be any positive integer.

7. An apparatus as in claim 1 wherein the input registers are adapted to convert the input serial data to an internal representation comprising separate sign, fraction and exponent.

8. An apparatus as in claim 1 wherein the memory means consists of read only memory for storage of constants and a read/write memory for storage of temporary results.

9. An apparatus as in claim 2 wherein the shifter/normalizer means is

adapted to perform binary weighted barrel shifting wherein the shifter function is determined by a control input to the shifter/normalizer and the normalizer function effects a data dependent shift of up to 15 bits within a single clock cycle.

10. An apparatus as in claim 2 wherein the arithmetic means implements logical operations such as but not limited to floating point addition, multiplication and multiply-accumulate algorithms using a parallel microcoded data path.

11. An apparatus as in claim 2 wherein the arithmetic means comprises a logical unit such as but not limited to an input-multiplexer, an adder, an output shifter, a flags unit and a control unit.

12. An apparatus as in claim 1 wherein the output register is adapted to be loaded in parts to enable the conversion from the internal representation to IEEE 754 floating point format and the output register is adapted to be parallel loaded from the arithmetic means or serially loaded from a serial source and the register is unloaded serially.

13. An apparatus as in claim 1 wherein the control and sequencing means includes timing and control logic, a microcode ROM, address decoders, branch control logic, flags logic, instruction register, instruction decoder and a program counter.

14. An apparatus as in claim 1 wherein there are three data buses, an X bus, a Y bus and a R bus and where the X and Y buses are called operand buses and the R bus is called the result bus.

15. An apparatus as in claim 1 wherein there is an accumulator comparison means.

16. A scalable array processor chip comprising an array of processing elements each said element including at least one input register means adapted to receive and process serial operands in the form of {instruction, data} 2-tuples, a memory means adapted to store temporary results and constants, a shifter/normalizer means adapted to shift or normalize data, an arithmetic means adapted to perform logical operations such as but not limited to addition, subtraction and partial multiplication operations, an output register means

adapted to output results from the processing element, a control and sequencing means adapted to control the operation of the processing element, a plurality of data buses adapted to provide communication between the plurality of means and wherein each processing element has means for communication only with adjacent elements.

17. An apparatus as in claim 16 wherein the array of elements comprise an interconnected lattice of at least one dimension.

18. An apparatus as in claim 16 wherein the array of elements comprise an interconnected lattice of at least two dimensions.

19. An apparatus as in claim 16 wherein the scalable array processor chip is adapted to perform at least the functions of computing the product of two or more matrices, computing the element-wise product of two or more matrices, computing the sum of two or more matrices, permuting the rows and columns of a matrix and transposing a matrix.

20. A computing apparatus comprising a host processor, at least one scalable array processor chip and a plurality of data formatters wherein the scalable array processor chip(s) and plurality of data formatters are adapted to perform matrix operations otherwise performed by the host processor.

21. An apparatus as in claim 20 wherein the apparatus includes a memory cache adapted to store operand data and temporary or intermediate results.

22. A method of performing matrix operations comprising the steps of :

(a) providing a plurality of processing elements in the form of an array adapted to perform systolic processing operations;

(b) receiving operand matrix data for processing from a host or data source;

(c) formatting the operand matrix data in a data formatter by adding an instruction to form an {instruction, data} 2-tuple;

(d) transferring sets of 2-tuples to the processing element array to cause the processing elements to process the data in accordance with the instruction;

(e) repeating the steps (b) to (d) a number of times said number of times being dictated by the matrix operation being performed;

(f) unloading the results of the matrix operations into an output result register of the processing elements (under control of an instruction specified by

the operand 2-tuple);

(g) transferring the contents of the output result registers held within the plurality of processing elements back to data formatters as result wavefronts;

(h) storing the result wavefront data back to a host or data sink; and

(i) repeating the steps (f) to (h) a number of times said number of times being dictated by the matrix operation being performed.

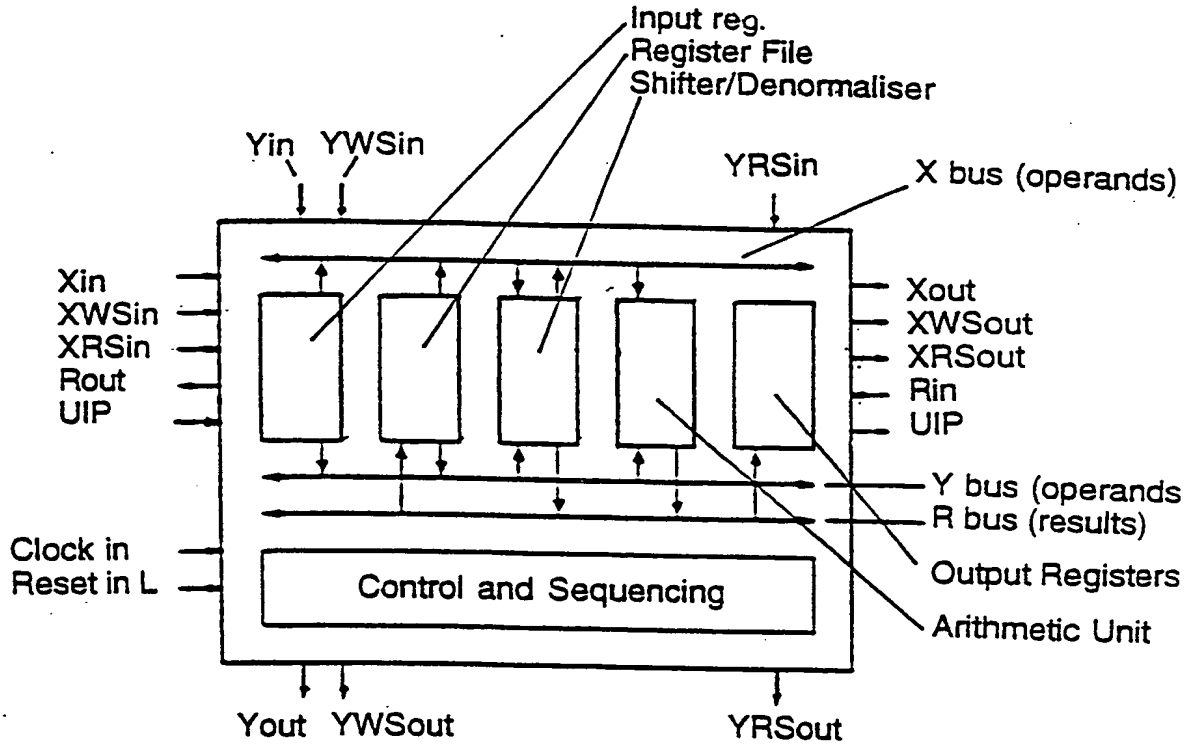

Input reg.
Register File
Shifter/Denormaliser
Yin
YWSin
YRSin
X bus (operands)
Xin
XWSin
XRSin
Rout
UIP
Xout
XWSout
XRSout
Rin
UIP
Y bus (operands
R bus (results)
Clock in
Reset in L
Control and Sequencing
Output Registers
Arithmetic Unit
Yout
YWSout
YRSout

FIG 1

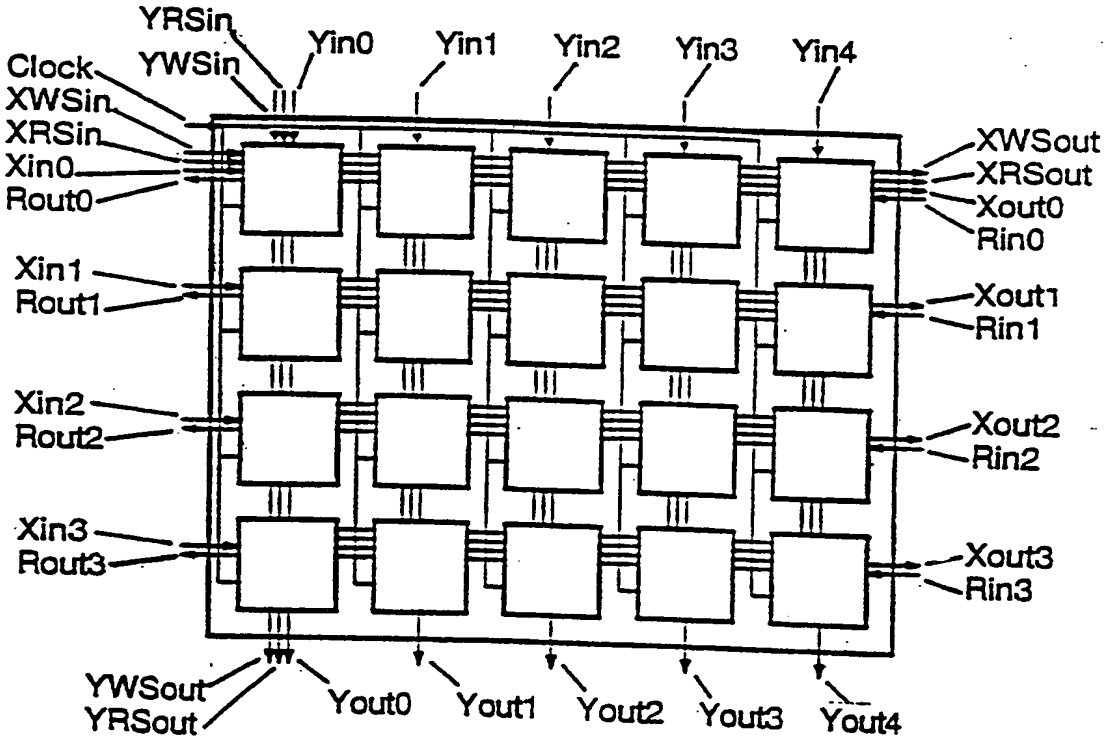

YRSin
YWSin
Yin0
Yin1
Yin2
Yin3
Yin4
Clock
XWSin
XRSin
Xin0
Rout0
Xin1
Rout1
Xin2
Rout2
Xin3
Rout3
XWSout
XRSout
Xout0
Rin0
Xout1
Rin1
Xout2
Rin2
Xout3
Rin3
YWSout
YRSout
Yout0
Yout1
Yout2
Yout3
Yout4

FIG 2

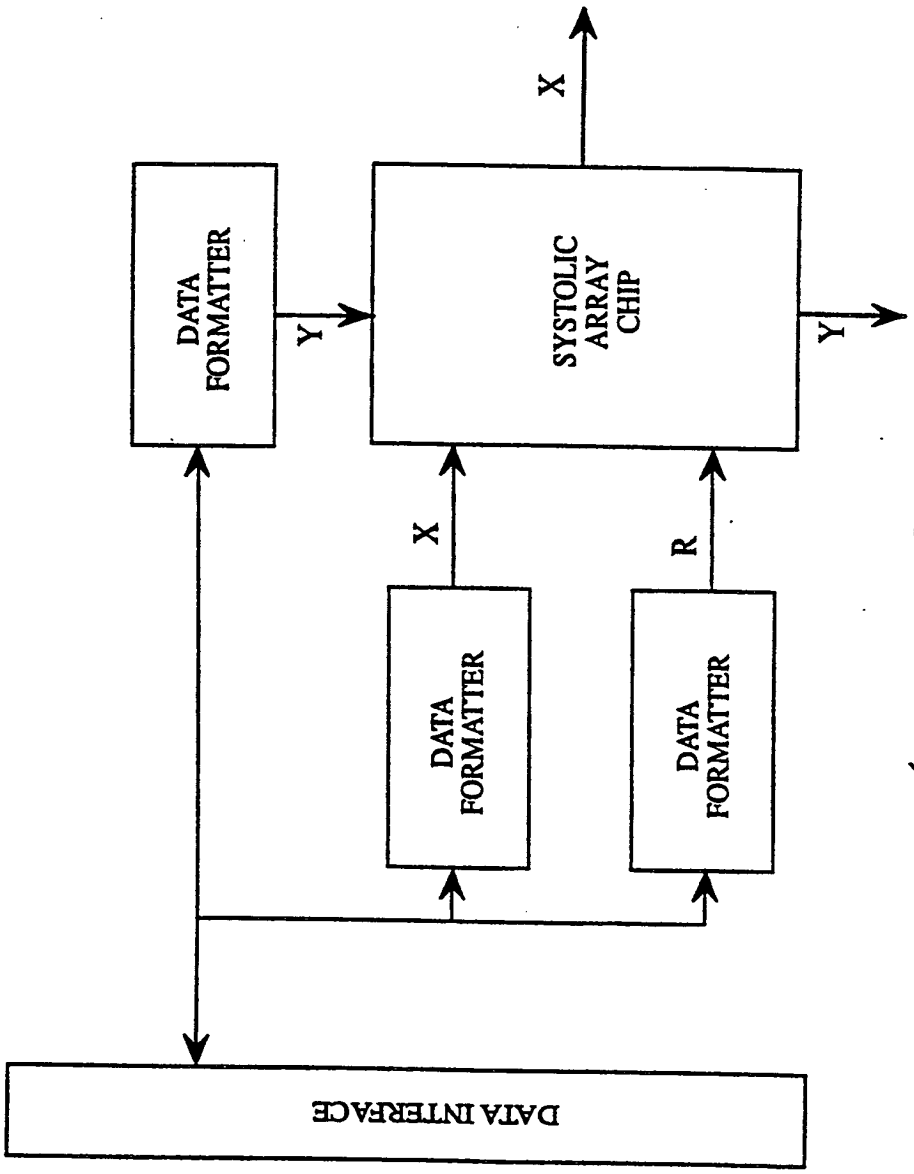

DATA INTERFACE
DATA FORMATTER
DATA FORMATTER
DATA FORMATTER
SYSTOLIC ARRAY CHIP
X
R
X
Y
Y
X

FIG 1

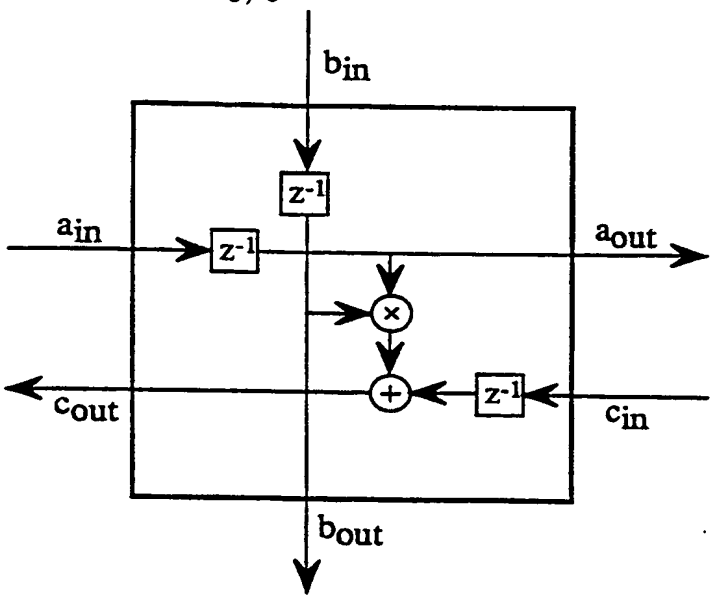

bin
z-1
ain
z-1
aout
cout
z-1
cin
bout

FIG 4

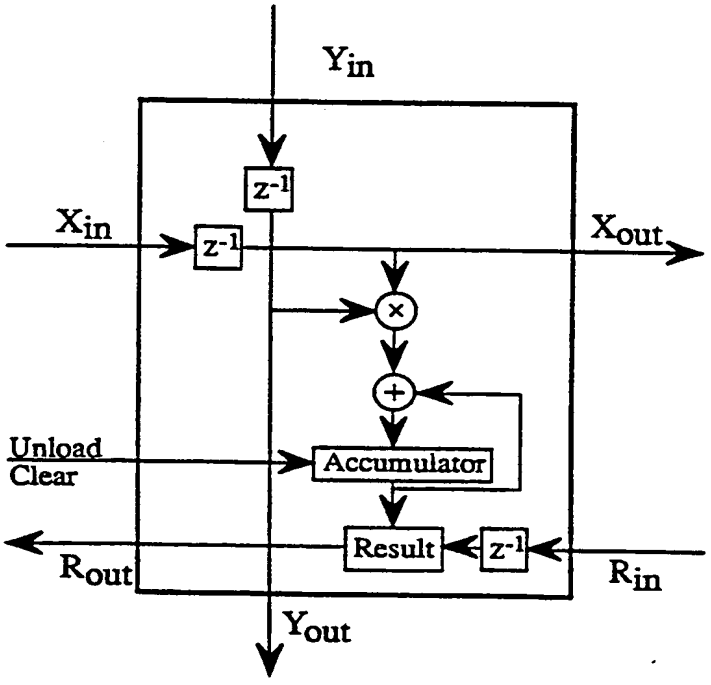

Yin
z-1
Xin
z-1
Xout
Unload
Clear
Accumulator
Result
z-1
Rout
Rin
Yout

FIG 5

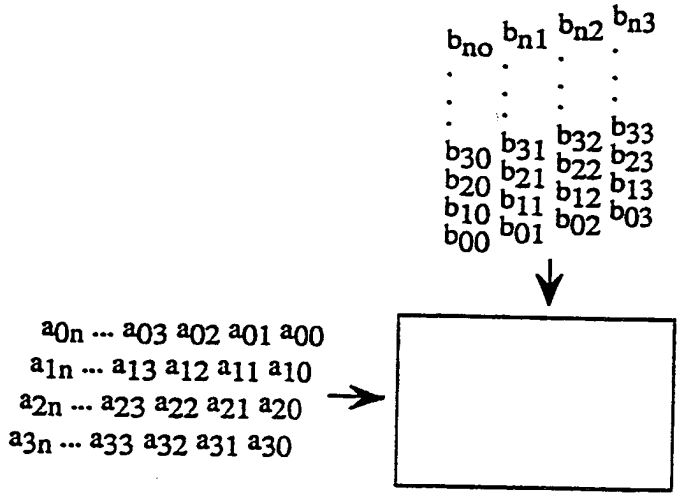

b_{no} b_{n1} b_{n2} b_{n3}
b_{30} b_{31} b_{32} b_{33}
b_{20} b_{21} b_{22} b_{23}
b_{10} b_{11} b_{12} b_{13}
b_{00} b_{01} b_{02} b_{03}
a_{0n} ... a_{03} a_{02} a_{01} a_{00}
a_{1n} ... a_{13} a_{12} a_{11} a_{10}
a_{2n} ... a_{23} a_{22} a_{21} a_{20}
a_{3n} ... a_{33} a_{32} a_{31} a_{30}

FIG 6

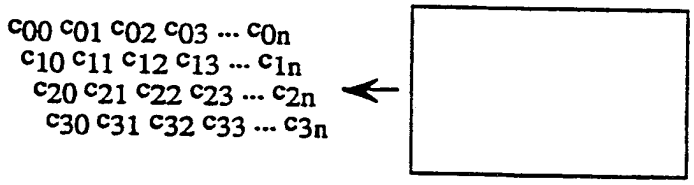

c_{00} c_{01} c_{02} c_{03} ... c_{0n}
c_{10} c_{11} c_{12} c_{13} ... c_{1n}
c_{20} c_{21} c_{22} c_{23} ... c_{2n}
c_{30} c_{31} c_{32} c_{33} ... c_{3n}

FIG 7

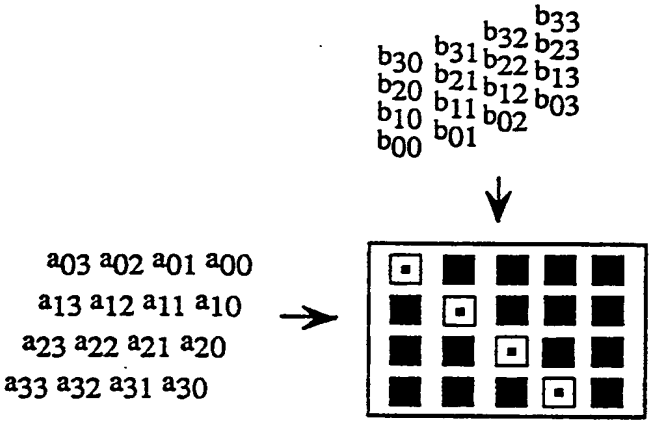

b33
b32 b23
b30 b31 b22 b13
b20 b21 b12 b03
b10 b11 b02
b00 b01
a03 a02 a01 a00
a13 a12 a11 a10
a23 a22 a21 a20
a33 a32 a31 a30

FIG 8

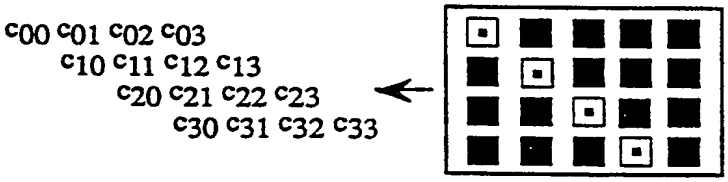

c00 c01 c02 c03
c10 c11 c12 c13
c20 c21 c22 c23
c30 c31 c32 c33

FIG 9

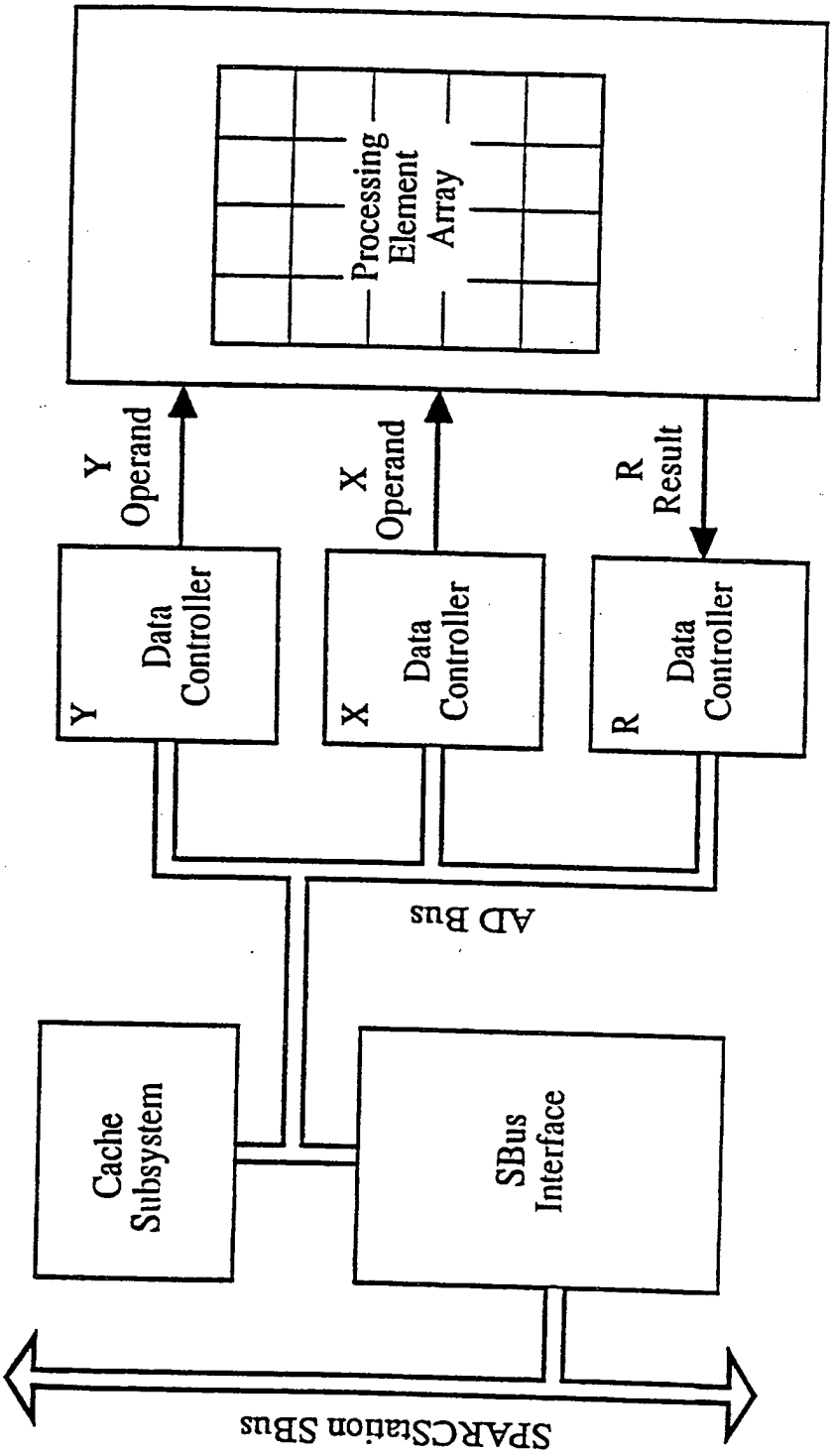

Processing Element Array
Y Operand
X Operand
R Result
Y Data Controller
X Data Controller
R Data Controller
AD Bus
Cache Subsystem
SBus Interface
SPARCStation SBus

FIG 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU 93/00573

A. CLASSIFICATION OF SUBJECT MATTER
Int. Cl.[5] G06F 15/347

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC G06F 15/34, 15/347, 7/50, 7/52

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
US Class : 364/232.21, 364/748, 364/754, 364/937.4
AU : IPC as above

Electronic data base consulted during the international search (name of data base, and where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to Claim No.
A	US,A, 5095527 (URAMOTO et al) 10 March 1992 (10.03.92) See the whole document, especially claim 1.	1-22
A	US,A, 4933895 (GRINBERG et al) 12 June 1990 (12.06.90) See column 3, lines 5 to 62.	1-22
A	US,A, 4686645 (McCANNY et al) 11 August 1987 (11.08.87) See column 1, lines 39 to 68 and claim 1.	1-22
A	US,A, 4543642 (HANSEN) 24 September 1985 (24.09.85) See the whole document, especially column 5, line 53 to column 6, line 57.	1-22

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 February 1994 (14.02.94)	17 FEB 1994 (17.02.94)
Name and mailing address of the ISA/AU AUSTRALIAN INDUSTRIAL PROPERTY ORGANISATION PO BOX 200 WODEN ACT 2606 AUSTRALIA Facsimile No. 06 2853929	Authorized officer John Thomson J W THOMSON Telephone No. (06) 2832214

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member					
US	5095527	JP	2054383				
US	4933895	DE JP	3873059 2500698	EP WO	322449 8900733	IL	86757
US	4686645	GB	2144245				
							END OF ANNEX